

Index Optimization with Combination of Text Clustering using String Vector based KNN and AHC Algorithm

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we propose the string vector based KNN algorithm and the sting vector AHC algorithm for automating the index optimization based on the text clustering. In the previous work, although the string vector based KNN version was proposed as an approach to the index optimization, a domain should be presented as an input text tag manually. In this research, texts are clustered into subgroups based on their contents, and each word in a novice text is classified into one of the three predefined categories, by nominating a classifier based on the similarity with the clusters. The AHC algorithm and the KNN algorithm which processes string vectors, instead of numerical vectors, are adopted respectively to the text clustering and the index optimization. The goals of this research are to automate the keyword extraction and to improve the performance of both tasks, using the string vector-based algorithms.

I. INTRODUCTION

The index optimization refers to the process of optimizing the index by both the index expansion and the index filtering. It is necessary to define the important classes, to decide whether to apply the index expansion or the index filtering to each word. The important classes are defined as the three classes: expansion, inclusion, and removal, and the index optimization is interpreted into a classification task where each word is classified into one of the three classes. The index expansion is applied to words which are classified into expansion, and the index filtering is applied to ones which are classified into removal. This research is interested in the classification; the actions which are taken after classifying words are excluded.

This research is motivated by the necessity of defining domains for designing the index optimization system. The index optimization is interpreted into classification tasks, as mentioned above; the index optimization system is composed with independent classification systems as many as domains. The process of designing the index optimization system is to predefine domains, collect sample examples domain by domain, and allocate a classifier for each domain. Another motivation of this research is the successful results from applying the string vector based KNN algorithm and the string vector based AHC algorithm respectively to the index optimization and the text clustering. This research is

intended to automate the domain decision by connecting the index optimization with the text clustering.

The idea of this research is to apply the string vector based KNN algorithm and the string vector based AHC algorithm to the index optimization and the text clustering which relate to each other. In this research, we initially prepare the text collection where each text is associated with words which are labeled with one of the three important levels. The semantic similarity between two string vectors is defined as a semantic operation on them, and the KNN algorithm and the AHC algorithm are modified into the string vector-based versions. A collection is segmented into clusters by the text clustering, and a cluster is viewed as a domain where a classifier for the index optimization is implemented independently. A domain is automatically decided to an input text by its similarities with the cluster prototypes.

Let us mention some benefits from this research. Words and texts are encoded into string vectors as the solution to the main problems in encoding them into numerical vectors. The performances in the index optimization and the text clustering by adopting the string vector-based versions of the KNN algorithm and the AHC algorithm. It is not necessary to predefine domains manually depending on prior knowledge or subjective views by automating it through the text clustering. It is not necessary to present a domain as a tag of an input text by automatically deciding a domain by the similarities with the cluster prototypes.

Let us mention some remaining tasks as the further discussions on this research. We modify more advanced machine learning algorithms and deep learning algorithms into the string vector-based versions. It is necessary to define and characterize mathematically more operations on string vectors. We implement the index optimization system which relates with the text clustering as a real program or a library. We need to install the index optimization in the text classification system or the text clustering system for improving their performances.

II. PROPOSED APPROACH

A. Similarity Metric

This section is concerned with the string vector as the representation of a word and the similarity between string vectors. The string vector is defined as a finite ordered set of strings; in the vector, the numerical values are replaced by the strings. It is required to define a similarity matrix for computing the similarity between elements. The similarity between string vectors is the average over one-to-one similarities between their elements. This section is intended to describe the process of encoding a word into a string vector and the process of computing the similarity between string vectors.

The process of encoding words into string vectors is illustrated in Figure 1. In the word-text association, a text collection is constructed by gathering texts, and each word is associated with its own texts based on their information. The string vector features are defined as symbolic forms manually in the feature definition. In the feature value assignment, the string vector which represents a word is filled with text identifiers which correspond to the features. Refer to [1] for studying the encoding process in more detail.

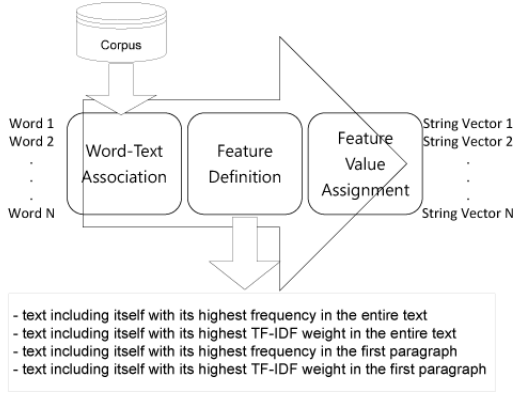


Figure 1. Process of Encoding Words into String Vectors

The similarity matrix which is the basis for computing the similarity between string vectors is illustrated in figure 00. In the matrix, each row and each column correspond to text identifiers, and each element is the similarity between texts, d_i and d_j , as shown in equation (1),

$$s_{ij} = \text{sim}(d_i, d_j) \quad (1)$$

The similarity between two texts, d_i and d_j , is computed by equation (2),

$$\text{sim}(d_i, d_j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|} \quad (2)$$

where D_i is the set of words in the text, d_i , and D_j is the set of words in the text, d_j . The value of similarity

between texts, d_i and d_j , is always given as a normalized value between zero and one, as shown in equation (3)

$$0 \leq \text{sim}(d_i, d_j) \leq 1 \quad (3)$$

The similarity matrix is used for computing the similarity between string vectors which represent words as a reference table.

$$\begin{bmatrix} s_{11} & s_{12} & \dots & s_{1N} \\ s_{21} & s_{22} & \dots & s_{2N} \\ \dots & \dots & \dots & \dots \\ s_{N1} & s_{N2} & \dots & s_{NN} \end{bmatrix} \quad \text{sim}(d_i, d_j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|}$$

$$0 \leq \text{sim}(d_i, d_j) \leq 1.0$$

Figure 2. Semantic Similarity Matrix

Let us mention the process of computing the similarity between string vectors as a semantic similarity between words. The two string vectors which represent words are notated by $\text{str}_1 = [d_{11} \ d_{12} \ \dots \ d_{1d}]$ and $\text{str}_2 = [d_{21} \ d_{22} \ \dots \ d_{2d}]$. The similarity between two string vectors is computed by equation (4),

$$\text{sim}(\text{str}_1, \text{str}_2) = \frac{1}{d} \sum_{i=1}^d \text{sim}(d_{1i}, d_{2i}). \quad (4)$$

The similarity between elements, $\text{sim}(d_{1i}, d_{2i})$, is taken from the similarity which was mentioned above. The value which is generated by equation (4) is always given as a normalized value between zero and one as shown in equation (5),

$$0 \leq \text{sim}(\text{str}_1, \text{str}_2) \leq 1. \quad (5)$$

Let us make some remarks on the process of encoding words into string vectors and computing the similarity between words. A word is encoded into a string vector by the word-text association, the feature definition, and the feature value assignment. The similarity matrix is defined as the basis for computing the similarity between string vectors. The similarity is computed by equation (4). In the future research, we consider representing a word into multiple string vectors.

B. String Vector based KNN Algorithm

This section is concerned with the string vector based KNN algorithm. It was initially proposed as the approach to the text summarization by Jo in 2018 ???. The similarity metric between string vectors which was described in the previous section is used for computing the similarity between a novice string vector and a training example. The labels of its nearest neighbors are voted with their identical weights for deciding the label of a novice input. This section is intended to describe the initial version of KNN algorithm from which its variants are derived.

The process of computing the similarity between a novice item and a training example is illustrated in Figure 00. The labeled words which are gathered as samples are encoded into string vectors, and they are notated by a set, $Tr = \{\mathbf{str}_1, \mathbf{str}_2, \dots, \mathbf{str}_N\}$. A novice word is encoded into a string vector notated by $\mathbf{str}$. Its similarities with the string vectors which represent the sample words are computed by equation (4), as $\text{sim}(\mathbf{str}, \mathbf{str}_1), \text{sim}(\mathbf{str}, \mathbf{str}_2), \dots, \text{sim}(\mathbf{str}, \mathbf{str}_N)$. Some training examples which are most similar as the novice input are selected as its nearest neighbors.

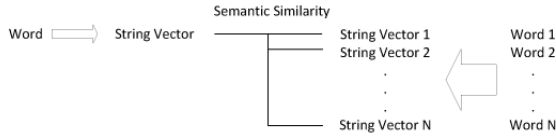


Figure 3. Similarities of Novice Input with Training Examples

In Figure 4, the process of selecting nearest neighbors by the ranking selection is illustrated. The training examples are sorted by the descending order of their similarities, after computing their similarities with a novice example. The training examples with their higher similarities with a novice example are selected as its nearest neighbors, as expressed in equation (6),

$$Nr = \text{Select}_k(Tr, \mathbf{str}), Nr \subseteq Tr \quad (6)$$

The set of nearest neighbors, Nr , is composed with elements as expressed in equation (7),

$$Nr = \{\mathbf{str}_1^{near}, \mathbf{str}_2^{near}, \dots, \mathbf{str}_k^{near}\} \quad (7)$$

The elements in the set, Nr , are used for voting their labels in the next step.

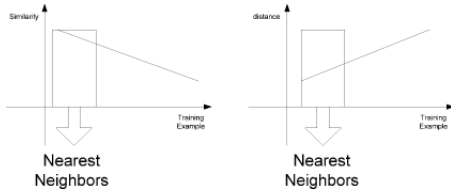


Figure 4. Selection of Most Similar or Least Distant Training Examples as Nearest Neighbors

The process of voting the labels of nearest neighbors for decoding the label of a novice word is illustrated in Figure 5. The predefined categories are notated by $c_1, c_2, \dots, c_m$, and the label of a nearest neighbor is notated by $c_j = \text{label}(\mathbf{str}_i^{near})$. The number of nearest neighbors which belong to the category, c_j , is notated by $\text{Count}(Nr, c_j)$. The label of the novice item, $\mathbf{str}$, is decided by equation (8),

$$\text{label}(\mathbf{str}) = \arg\max_{j=1}^m \text{Count}(Nr, c_j) \quad (8)$$

The process of deciding the label of a novice item by equation (8), is called voting.



Figure 5. Voting of Labels of Nearest Neighbors for Deciding Label of Novice Input

Let us make some remarks on the string vector based KNN algorithm. It computes the similarities of a novice item with the training examples. The training examples with their highest similarities with the novice item are selected as its nearest neighbors. The label of the novice item is decided by voting the labels of its nearest neighbors. The ranking selection is adopted for selecting the nearest neighbors from training examples, in this version.

C. System Architecture

This section is concerned with the index optimization system which adopts the string vector based KNN algorithm. In Section II-B, we described the proposed version of KNN algorithm as the approach to the index optimization. It is viewed as a classification task which classifies a word into one of the three categories. This section is intended to describe the index optimization system with respect to its function and its architecture.

The process of collecting the same words and encoding them into string vectors for implementing the index optimization system is illustrated in Figure 6. The index optimization is interpreted into a domain dependent classification; even a same word is classified differently, depending on the domain. For each domain, an independent classification task is defined. The several domains are decided, and the words which are labeled with one of the three categories exclusively in each domain. It is required to present the domain from a text which is given as the input for doing this task.

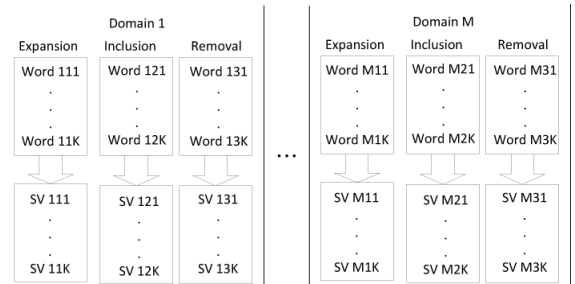


Figure 6. Preparation of Training Examples

The entire architecture of the proposed index optimization system is illustrated in Figure 7. A text is given as the input, and words are extracted from it in the indexing

The diagram illustrates the architecture of the proposed system. It starts with 'Text Collection' leading to 'Attributes' and 'Input Text'. The 'Input Text' is processed by a 'Text Indexer' to produce 'Word' and 'Unlabeled' outputs. The 'Attributes' are processed by a 'Feature Discriminator'. The 'Input Text' is also processed by an 'Encoder' (Word $\rightarrow$ String Vector). The 'Feature Discriminator' and 'Encoder' both output 'Word' and 'Unlabeled' results. These results are then processed by a 'Voting Module' and a 'Nearest Neighbor Discriminator' to produce 'Classification' and 'Nearest Neighbors' outputs. A 'Training Example Discriminator' and 'Similarity Computation Module' (Semantic Similarity) are also shown, which take 'Attributes' and 'Input Text' as input and output 'Similarity Matrix' and 'Nearest Neighbors'.

Extension Group	Inclusion Group	Removal Group
Word	Word	Word
Word	Word	Word
.....		
Word	Word	Word

Input Text

Text Indexer

Word

Unlabeled

Classification

Attributes

Feature Discriminator

Encoder

Word $\rightarrow$ String Vector

Voting Module

Nearest Neighbor Discriminator

Text Collection

Similarity Matrix

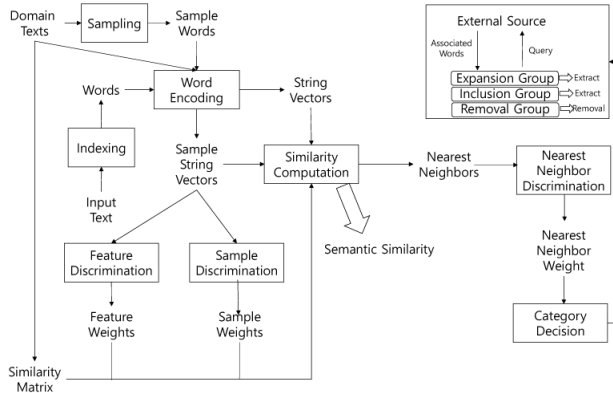
Training Example Discriminator

Similarity Computation Module

Semantic Similarity

Nearest Neighbors

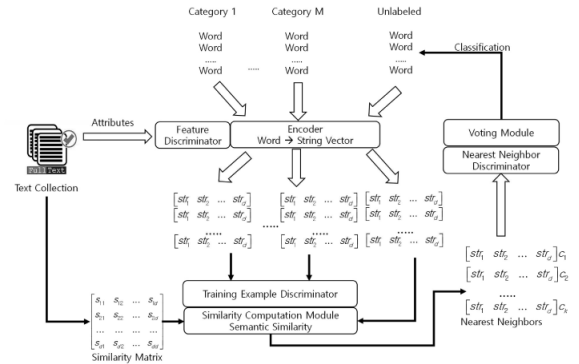
The execution process of the proposed system is illustrated as a block diagram in Figure 8. Sample words which are labeled with one of the three categories are collected within a domain, and they are encoded into string vectors. An input text is indexed into a list of words, and they are also encoded into string vectors. The nearest neighbors are selected by the similarity computation, and the label is decided by voting ones of their nearest neighbors for each word. Ones which are classified into expansion require their associated words from external sources, ones which are classified into inclusion are preserved, and ones which are classified into removal are excluded.



Let us make some remarks on the proposed system which is illustrated in Figure 7 as its architecture. The index optimization is defined as a multiple classification task

D. Connection with Text Clustering

The architecture of the text clustering system which was proposed in [3] is illustrated in Figure 9. The texts in the group are encoded into string vectors, and the AHC algorithm which is proposed in [3] is adopted as the approach. It starts with singletons as many as items, computes the similarities of all possible cluster pairs, and merge the cluster pair with its highest similarity into one cluster. The two steps are iterated until the number of clusters reach the desired number. We may consider replacing the AHC algorithm by its variants for improving the speed and the performance.



The connection of the index optimization with the text clustering is illustrated in Figure 10. The M domains are defined by clustering texts in a group, initially and automatically. A novice text is given as the input, and its domain, domain D, is decided by its similarities with domains. A classifier which corresponds to domain D is nominated and classify each word in a novice text into expansion, inclusion, or removal. The M index optimization systems are viewed

as independent ones, each of which classifies each word into one of the three categories.

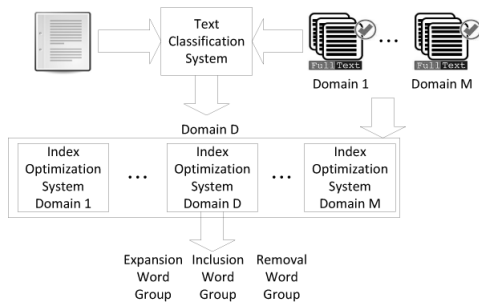


Figure 10. Index Optimization System connected with Text Clustering

The execution flow of index optimization which is connected with the text clustering is illustrated in Figure 11. Unlabeled texts are clustered into subgroups, each of which is a domain. Sample words which are labeled with expansion, inclusion, or removal are generated from each domain. These sample words are provided for training the classifier in each index optimization system. Each classifier is used for judging the importance level of each word which is indexed from the input text.

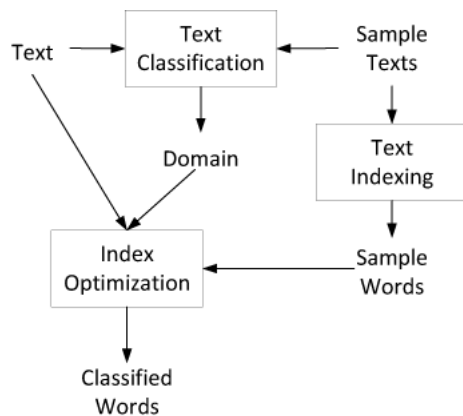


Figure 11. System Flow of Index Optimization connected with Text Clustering

Let us make some remarks on the connection of the index optimization with the text clustering. It is the process of segmenting a text group into subgroups based on their content-based similarities. The role of text clustering is to define the domains initially in the architecture of connecting the index optimization with the text clustering. In each domain, sample words are generated, and its corresponding classifier is trained with them. In the future research, we consider using the index optimization for clustering texts in the opposite direction.

REFERENCES

- [1] T. Jo, "Automatic Text Summarization using String Vector based K Nearest Neighbor", 6005-6016, Journal of Intelligent and Fuzzy Systems, 35, 2018.
- [2] T. Jo, "String Vector based Version of K Nearest Neighbor for Index Optimization in Current Affairs", 47-50, The Proceedings of International Conference on Applied Cognitive Computing, 2018.
- [3] T. Jo, "Semantic String Operation for Specializing AHC Algorithm for Text Clustering", 1083-1100, Annals of Mathematics and Artificial Intelligence, 2019.